Úvod do automatizace správy serverů s Ansible

Lukáš Novák

55. konference EurOpen
15.5.2023

Představení

- alma mater
  - Katedra informatiky na Univerzitě Palackého v Olomouci
- zakladatel vývojářského studia beecode (www.beecode.io)
- vývoj aplikačního serveru v Node.js, správa infrastruktury
- vedení produktu Unizone
  - mobilní aplikace pro studenty a vyučující na vysokých školách





Motivace

- co nejrychleji připravit server/y pro spuštění produkční aplikace
  - při prvotní instalaci
  - při potřebě horizontálně škálovat
  - ...
- možnosti?
  - a) ručně přes terminál
  - b) ručně přes interaktivní „klikací“ nástroj
  - c) využití cloudových služeb (např. managed Kubernetes)
  - **d) použití IaC sofware (Ansible, Chef, Puppet, ...)**

Infrastructure as a Code (IaC)

„Infrastruktura jako kód (IaC) je proces správy a vytváření počítačových datových center pomocí **strojově čitelných definičních souborů** místo fyzické konfigurace hardwaru nebo interaktivních konfiguračních nástrojů.“

~ možnost zacházet s infrastrukturou jako s kódem

Ansible

- open source sada nástrojů a modulů pro **Infrastructure as a Code**
- Ansible vyvíjí tzv. „Ansible komunita“ (a Red Hat Inc.)

- architektura:
  - dva typy strojů: řídicí a uzly
  - bez daemona, na uzlech není třeba žádný speciální software

- minimální systémové nároky:
  - linuxové systémy i Windows
  - Python >= 2.4, OpenSSH (PowerShell)

Vlastnosti Ansible

- jednoduchost
  - je jednoduché rychle začít
- minimalismus
  - není třeba instalace dodatečného software na uzlech
- bezpečnost
  - pro komunikaci je využíváno praxí ověřené OpenSSH
- spolehlivost
  - správně napsané moduly nemají vedlejší efekty ani při opakovaném vykonání ~ idempotence

Idempotence

„Opakovaným vykonáním operace se stejným vstupem vznikne stejný výstup který vznikne jedním vykonáním.“

~ pokud např. při vykonání konfigurace dojde k výpadku proudu, opětovným vykonáním dostaneme systém do požadovaného stavu

Základní komponenty

- terminálová utilita
- inventory (kolekce cílů)
- modules (operace)
- playbooks (posloupnost aplikování modulů)

```
novalu@penguin:~$ ansible --version
ansible 2.10.8
  config file = None
  configured module search path = ['/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.9.2 (default, Feb 28 2021, 17:03:44) [GCC 10.2.1 20210110]
```

Inventory

- kolekce cílových uzlů
- většinou jde o IP adresy serverů, může jít i o síťové prvky, disková pole nebo např. adresy Docker kontejnerů
- inventáře mohou být dynamické a získávat cíle z externích zdrojů
- cíle mohou být sdružovány do skupin
- každý cíl může mít přiřazené proměnné

Inventory (příklad)

- pro účely demonstrace vytvoříme 4 stroje na cloudu DigitalOcean
- jde o čisté stroje se systémem Ubuntu 22.10
- soubor inventory.yaml obsahuje typ připojení, hostitele a přihlašovací údaje

inventory.yaml

```yaml
all:
  hosts:
    ubuntu1:
      ansible_connection: ssh
      ansible_host: 159.223.251.82
      ansible_user: root
      ansible_password: Europen2023Demo

    ...

    ubuntu4:
      ansible_connection: ssh
      ansible_host: 159.223.248.66
      ansible_user: root
      ansible_password: Europen2023Demo
```

Moduly

- zajišťují různé oblasti konfigurace, např. nastavení a management serveru, nastavení sítě, virtualizace, kontejnery, atd.
- v dokumentaci nyní cca 7382 modulů
- můžeme si vytvořit vlastní
- příklady základních modulů
  - ping
  - file
  - copy
  - command
  - ...

Vykonání operace modulu (modul ping)

- modul, pomocí kterého můžeme ověřit připojení k uzlům

```
novalu@penguin:~/Europen$ ansible all -m ping
ubuntu4 | SUCCESS => { "ping": "pong" }
ubuntu3 | SUCCESS => { "ping": "pong" }
ubuntu1 | SUCCESS => { "ping": "pong" }
ubuntu2 | SUCCESS => { "ping": "pong" }
```

Vykonání operace modulu (modul file)

- modul, pomocí kterého můžeme pracovat se soubory
- např. vytvoříme soubor pomocí utility touch

```
$ ansible all -m file -a 'path=/tmp/test state=touch'
```

```
ubuntu4 | CHANGED => { ... }
ubuntu1 | CHANGED => { ... }
ubuntu3 | CHANGED => { ... }
ubuntu2 | CHANGED => { ... }
```

Vykonání operace modulu (modul file)

- modul, pomocí kterého můžeme pracovat se soubory
- např. změníme oprávnění souboru na 600

```
$ ansible all -m file -a 'path=/tmp/test state=file mode=600'
```

první běh:

```
ubuntu4 | CHANGED => { ... }
ubuntu1 | CHANGED => { ... }
ubuntu3 | CHANGED => { ... }
ubuntu2 | CHANGED => { ... }
```

druhý běh:

```
ubuntu4 | SUCCESS => { ... }
ubuntu1 | SUCCESS => { ... }
ubuntu3 | SUCCESS => { ... }
ubuntu2 | SUCCESS => { ... }
```

Playbooks

- play
  - vykonání operace pomocí modulu
  - každé play je definováno množinou cílů a úkoly (tasks)
- playbook
  - znovupoužitelné soubory definující posloupnost play
  - formát YAML

Playbooks (příklad)

- 2x play
  - vytvoření souboru /tmp/test
    - modul file
    - argumenty path a state
  - nastavení oprávnění souboru
    - modul file
    - argumenty path, state a mode

playbook.yaml

```yaml
- name: Create test file
  hosts: all
  become: yes
  tasks:
    - file:
        path: /tmp/test
        state: touch

- name: Set permissions
  hosts: all
  become: yes
  tasks:
    - file:
        path: /tmp/test
        state: file
        mode: 600
```

Běh playbooku (příklad)

- playbook spustíme pomocí utility ansible-playbook

```
$ ansible-playbook playbook.yml
```

Příklad

1. aktualizace systému
2. vytvoření docker user group
3. instalace docker engine
4. instalace python docker balíčku
5. sestavení docker obrazu
6. spuštění docker obrazu
7. ???